Program Reasoning

16. Program Synthesis as AI

Kihong Heo

KAIST

Inductive Logic Programming (ILP)

- A subfield of symbolic artificial intelligence
- Goal: given a dataset (inductive), infer a set of rules (logic programming)
- Synthesizing programs in logic programming languages
    - E.g., Prolog, Datalog

Example

- Parent

Dataset

```
parent(a,b)    parent(a,c)    parent(d,b)
father(a,b)    father(a,c)    mother(d,b)
male(a)        female(c)      female(d)
```

Learned Rules

```
father(X,Y) :- parent(X,Y) & male(X)
mother(X,Y) :- parent(X,Y) & female(X)
```

- Transitive closure

Dataset

```
edge(1,2)    edge(2,3)    edge(3,4)
path(1,2)    path(1,3)    path(1,4)
path(2,3)    path(2,4)    path(3,4)
```

Learned Rules

```
path(X,Y) :- edge(X,Y)
path(X,Y) :- edge(X,Y) & path(X,Y)
```

Datalog Programs

- A set of Horn clause rules ($X_1 \wedge X_2 \wedge \cdots \wedge X_n \rightarrow H$)
- Input & output: a set of tuples
- Applications: big data analysis, network protocol, program analysis, etc

```
edge(1,2)
edge(2,3)
edge(3,4)
```

➡

```
path(X,Y) :- edge(X,Y)
path(X,Y) :- edge(X,Y) & path(X,Y)
```

➡

```
path(1,2)  path(1,3)
path(1,4)  path(2,3)
path(2,4)  path(3,4)
```

Datalog Program Synthesis

- Given a set of candidate rules, find a subset that is consistent with a given examples
  - A typical combinatorial optimization problem (i.e., NP-hard)
- In this lecture, we assume a set of candidates is predefined

```
edge(1,2)
edge(2,3)
edge(3,4)
```

I

```
✓ path(x,y) :- edge(x,y).
✗ path(x,x) :- edge(x,y).
✓ path(x,z) :- edge(x,y), path(y,z).
✗ path(x,y) :- path(y,x).
                …
```

O^+

```
path(1,2)  path(1,3)
path(1,4)  path(2,3)
path(2,4)  path(3,4)
```

O^-

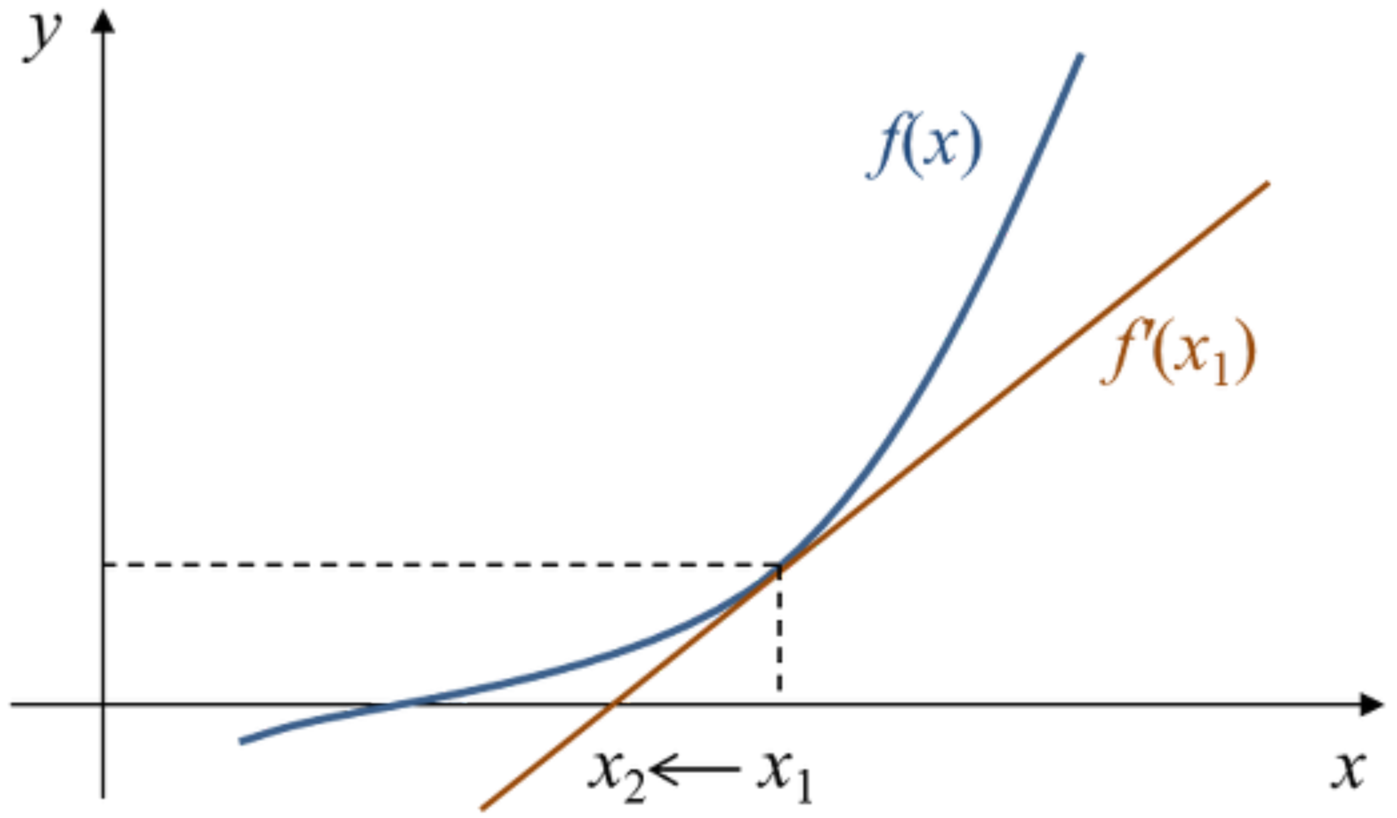
```
path(2,1)  path(3,1)
path(4,1)  path(4,2)
```

Challenges

- Huge search space
  - E.g., # of possible combinations of 50 candidate rules?
- If a wrong rule is chosen? The program produces a wrong tuple
- If a correct rule is missed? The program does not produce a correct tuple

A Solution: Difflog*

- A Datalog synthesis algorithm using numerical optimization
- Key idea: solving combinatorial optimization via numerical optimization
- Why numerical optimization? Many powerful algorithms exist!
  - E.g., Newton's method for differentiable loss functions
- Problem: Datalog programs are not differentiable



*Si et al., Synthesizing Datalog Programs using Numerical Relaxation, IJCAI 2019

Idea 1: Provenance

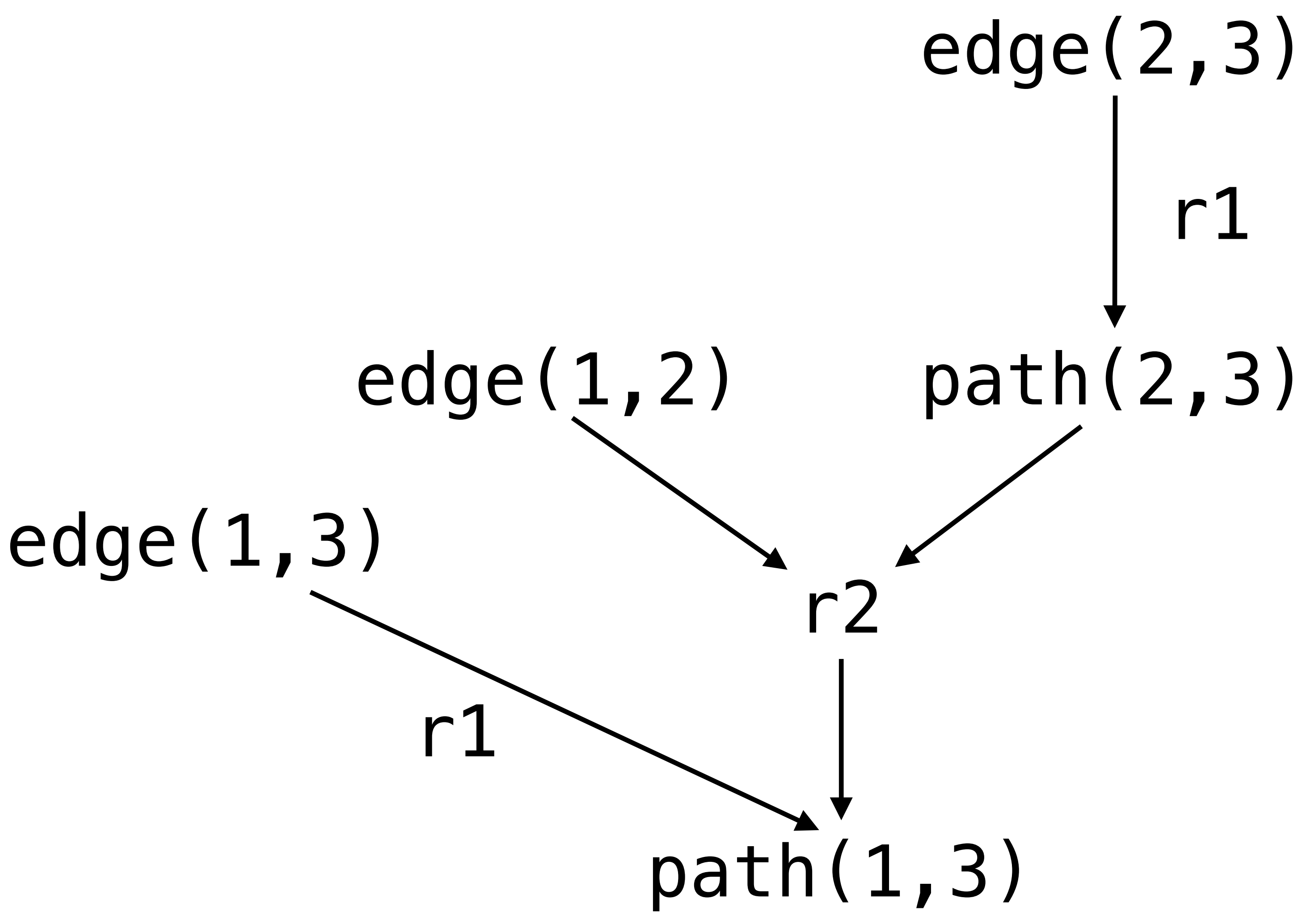
- Datalog programs produce provenance as well as output tuples
- Provenance: a proof that explains why a tuple is derived
  - If an undesired tuple is derived, we can see the reason

Rules

```
r1:  path(x,y) :- edge(x,y).
r2:  path(x,z) :- edge(x,y), path(y,z).
```

Input tuples

```
{edge(1,2), edge(2,3), edge(1,3)}
```



Idea 2: Continuous Semantics (1)

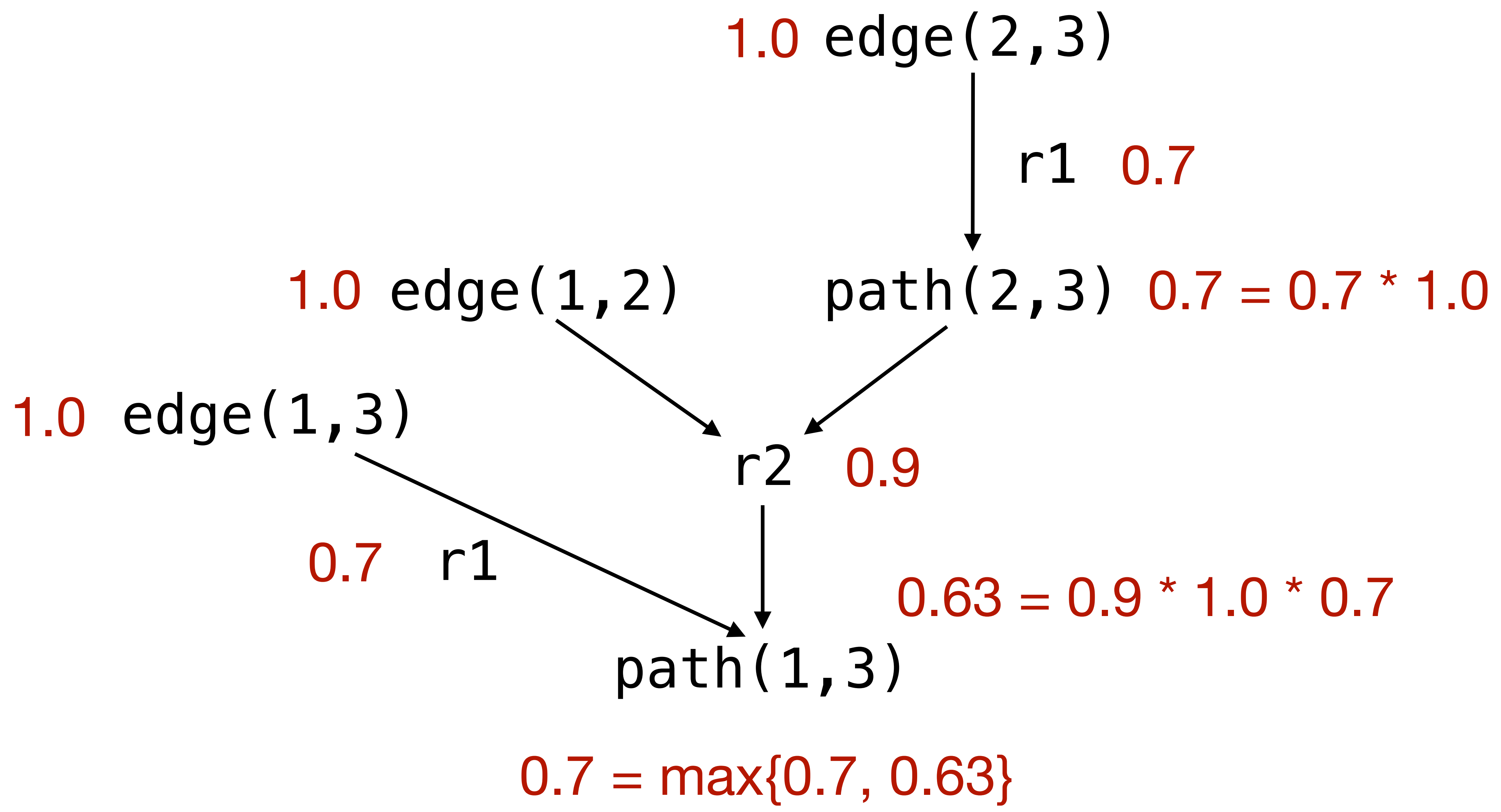
- Interpret Datalog programs (non-continuous function) as continuous functions
  - Existence of tuple {0, 1} to weight of tuple [0, 1]
  - Each rule is associated with a weight
  - The weight of a tuple is computed using the weights of rules on the provenance
- Then, combinatorial optimization problem → numerical optimization problem
  - Many existing algorithms applicable

Example (1)

Parameters: $\vec{W}$

```
0.7 path(x,y) :- edge(x,y).
0.9 path(x,x) :- edge(x,y).
0.1 path(x,z) :- edge(x,y), path(y,z).
0.3 path(x,y) :- path(y,x).
```

Input	edge(1,2)	edge(2,3)
Weight	1.0	1.0



Discrete semantics

$$v_t = \bigvee_g (v_{a_1} \wedge v_{a_2} \wedge \cdots \wedge v_{a_k})$$

Continuous semantics

$$v_t = \max_g (w_g \times v_{a_1} \times v_{a_2} \times \cdots \times v_{a_k})$$

Example (2)

Parameters: $\overrightarrow{W}$

```
0.7 path(x,y) :- edge(x,y).
0.9 path(x,x) :- edge(x,y).
0.1 path(x,z) :- edge(x,y), path(y,z).
0.3 path(x,y) :- path(y,x).
```

Input	edge(1,2)	edge(2,3)
Weight	1.0	1.0

Output	path(1,2)	path(2,3)	path(1,3)	path(1,1)	path(2,1)
Weight (v_t)	0.7	0.7	0.63	0.1	0.21
Expectation	1	1	1	0	0

Discrete semantics

$$v_t = \bigvee_g (v_{a_1} \wedge v_{a_2} \wedge \cdots \wedge v_{a_k})$$

Continuous semantics

$$v_t = \max_g (w_g \times v_{a_1} \times v_{a_2} \times \cdots \times v_{a_k})$$

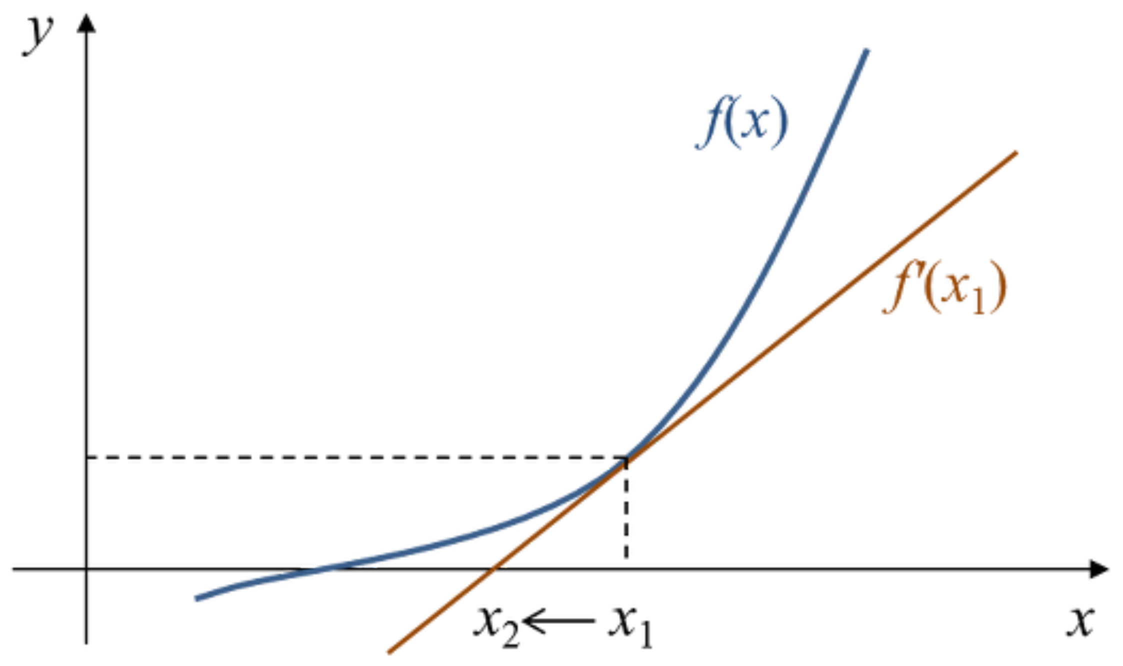
Optimization problem:
find $\overrightarrow{W}$ that minimizes the following loss

$$loss = \Sigma_{t \in pos}(1 - v_t)^2 + \Sigma_{t \in neg}(0 - v_t)^2$$

Numerical Optimization

$$loss = \Sigma_{t \in pos}(1 - v_t)^2 + \Sigma_{t \in neg}(0 - v_t)^2$$

- In continuous semantics, v_t is a polynomial with w_r (differentiable)
- Then, the loss function is a polynomial with w_r (differentiable)
- Solve using a well-known algorithm (e.g., Newton's method)
    - loss is 0 = consistent with all the examples = a desired program



Summary

- Inductive logic programming (ILP): symbolic AI $\cap$ program synthesis
- A long-standing argument in AI: connectionism vs symbolism
  - Connectionism (neural network): good at image recognition, machine translation, etc
  - Symbolism (logic): good at equation solving, logical reasoning, etc
- What is the future of AI (or programming)?
  - Neuro-symbolic AI: how to effectively combine both paradigms?